1984

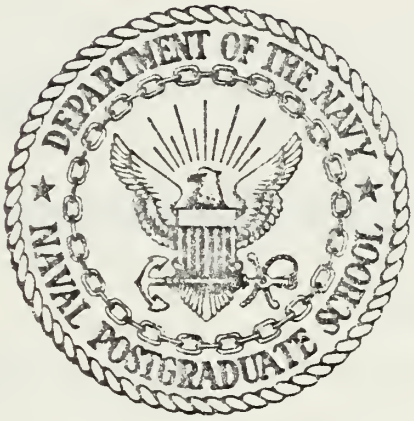
A preliminary DDS design for SPLICE based upon the TANDEM DBMS

Ruff, David C.; Johnson, James R.

Monterey, California. Naval Postgraduate School

DUDLEY KNOX [illegible]
NAVAL
M[illegible] [illegible]IA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

A PRELIMINARY DDS DESIGN FOR SPLICE
BASED UPON THE TANDEM DBMS

by

David C. Ruff
James R. Johnson

March 1984

Thesis Advisor: Dan Dolk

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Preliminary DDS Design for SPLICE based upon the TANDEM DBMS		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March 1984 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ruff, David C. Johnson, James R.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		12. REPORT DATE March 1984 13. NUMBER OF PAGES 78
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SPLICE, DDS, data dictionary/directory system, information resource management, database design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis considers a Data Dictionary/Directory System (DDS) for the Stock Point Logistics Integrated Communications Environment (SPLICE) project using the TANDEM DBMS package. The thesis first gives a background on SPLICE, then describes the concept of Data Dictionary/Directory Systems. In the coming age of information resource management, DDSs will increasingly become an important management tool of the database administrator. Highlights of the DDS facilities are mentioned as are design and distribution considerations.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A Preliminary DDS Design for SPLICE Based Upon the TANDEM DBMS

by

David C. Ruff
Lieutenant, United States Navy
B.B.A., University of Mississippi, 1976

and

James R. Johnson
Captain, United States Marine Corps
B.A., Central Washington University, 1975
M.B.A., National University, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1984

ABSTRACT

This thesis considers a Data Dictionary/Directory System (DDS) for the Stock Point Logistics Integrated Communications Environment (SPLICE) project using the TANDEM DBMS package. The thesis first gives a background on SPLICE, then describes the concept of Data Dictionary/Directory Systems. In the coming age of information resource management, DDSs will increasingly become an important management tool of the database administrator. Highlights of the DDS facilities are mentioned as are design and distribution considerations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. SPLICE AND THE DATA DICTIONARY/DIRECTORY SYSTEM

A. BACKGROUND

The purpose of the Stock Point Logistics Integrated Communications Environment (SPLICE) is to provide an environment to effectively and efficiently support the interactive and telecommunications requirements of all current and new projects operating within the Navy Uniform Automated Data Processing System for Stock Points (UADPS-SP). The Naval Supply Systems Command (NAVSUP) is the project sponsor for SPLICE with the UADPS-SP sites being the primary users along with other Department of Defense (DoD) / Navy systems. [Ref. 1]

NAVSUP initiated the SPLICE project to consolidate the telecommunications requirements of several different projects, in various stages of design, which will impact all of the Navy UADPS-SP sites [Ref. 2]. (See list of projects which follows this section.) Implementation of SPLICE is expected to provide the following capabilities:

- sign-on security
- CRT/terminal management
- screen formatting
- edit, validation and correction
- fault-tolerant operation
- data communications front end
- processing with menu capability
- interactive transaction entry
- access to database management systems

SPLICE will consolidate the telecommunications network with a standard suite of hardware and software. The SPLICE concept involves the use of a single minicomputer hardware

and software suite in conjunction with the existing UADPS-SP Burroughs system. The SPLICE minicomputers are intended to absorb and contain the communications handling workload that is currently forcing the host Burroughs CPU's into saturation. The SPLICE concept calls for standard minicomputers to be employed as foreground processors at each UADPS-SP site and interfaced via a high speed data network (HSDN) to the Burroughs medium size system. The foreground minicomputer will handle communication lines and terminal management, support interactive operations and stage messages for the background processors. The Burroughs background systems will handle large file processing applications, report preparation, batch processing and data base management functions.

SPLICE will utilize a standard minicomputer and a modular software design. A modular set of embedded computer software components will provide flexibility in that the configuration of each site within the SPLICE network can be designed to meet its unique requirements. The SPLICE configuration will consist of multiple processors with shared resources. A SPLICE configuration may interface with one or more mainframes, function as a satellite off another SPLICE complex, or function as a stand alone system [Ref. 3]. Within the SPLICE network, each terminal and computer process in the system can potentially access any other terminal and computer process.

A common element within the SPLICE system is the use of CRT's to provide interactive processing capabilities for a "user oriented system." The CRT display terminals will interact with the application logic and fetch information from system data bases. The current stock point system does not support interactive processing nor does it have the capacity to add a major increase in terminal support or associated processing requirements.

The design philosophy reinforced throughout the SPLICE project will be device independent management capability. No application program will read or write directly to or from a particular device. The driving force behind the SPLICE project is the need to provide computer and communications interfaces for the many application initiatives which cross boundaries of tasking and funding of many Navy major claimants [Ref. 2]. Projects that are to be tied to UADPS-SP under the SPLICE project are:

- Automation of Procurement and Accounting Data Entry (APADE)
- Centralized Accounting and Billing (CAB)
- Closed Loop Aeronautical Management Program (CLAMP)
- Disk Oriented Supply System (DOSS)
- Fixed Allowance Management and Monitoring System (FAMMS)
- Financial Improvement Program (FIP)
- Integrated Disbursing and Accounting (IDA)
- Multiple Activity Processing System (MAPS)
- Management Information System International Logistics (MISIL)
- Navy Automated Transportation System (NATDS)
- Navy Automated Transportation Documentation System (NAVADS)
- Navy Standard Civilian Payroll System (NAVCIPS)
- On Line Autodin (OLA)
- Operating Target Accounting for TRIDENT (OPTAR)
- Receipt Improvement Project (RIP)
- TRIDENT submarine Logistics Data System (TRIDENT LDS)
- Requisition Monitoring and Material Expediting (RM&ME)

B. NATURE OF THE PROBLEM

Research and thesis work on the SPLICE project thus far has primarily emphasized telecommunications and local area network (LAN) design within the SPLICE project [Ref. 4]. Because of the nature of the project and the extent of its future impact on other DoD/Navy systems, SPLICE should also be looked at as an environment which calls for the application of a Data Dictionary/Directory System (DDS) to fullfill its expectations.

C. OBJECTIVE

TANDEM has been selected as the minicomputer of choice for the SPLICE project. As a result, the objective of this thesis to suggest a preliminary DDS design for SPLICE based upon the TANDEM DBMS.

D. METHODOLOGY

This thesis will look at the current state of the SPLICE DDS and what an active DDS can do for the SPLICE project. It will look at methods to evaluate a DDS and apply these to the DDS design. DDS design considerations within the SPLICE environment will be identified, such as which resources need to be represented in the DDS, and to the extent possible, integrated into the recommended proposal. In the process, questions on the distribution of the DDS within SPLICE will also be addressed with recommendations for selection of the format which is most beneficial to the objectives of the SPLICE project.

II. DATA DICTIONARY/DIRECTORY SYSTEM CONCEPTS AND CONSIDERATIONS

A. INTRODUCTION

Data is a very useful and necessary resource to an organization. Data is a general term used to express any or all facts, numbers, letters, and symbols that refer to or describe an object, idea, condition, situation or other factor. It can be used to influence management decisions, by providing the manager with timely and accurate information. With these thoughts in mind, it is very important that data as a resource be easily accessible for proper and effective management.

The early design of Navy data processing systems revolved around specific application systems (UADPS-SP), hence the data was organized so that it would be machine and application specific. Therefore the data seldom crossed operational, functional or organizational boundaries. The outcome of this early design was multiple definitions of the same data as individual/independent data files were generated creating much redundancy and overlap.

As the role of the computer has grown in the Navy Supply System, the need/requirement for system integration has become evident, especially in the area of data. The introduction of Database Management Systems (DBMS) solves many of the information resource problems by organizing the data elements under consistent controls and structures, by providing a single flexible facility for accommodating different data files and operations, and demanding less programming effort than conventional programming languages.

This centralization of control is growing in acceptance and the function of implementing this centralized control is in the hands of the Database Administrator (DBA) [Ref. 5]. The data administration function is responsible for the system wide inventory and control of data. It requires pertinent facts and relationships about the varying data elements, processes, users and equipment to be located in the data-processing environment [Ref. 6]. Data administration internal control procedures restrict access to data, manage development of new types of data, and protect data from erroneous update and system failure.

A software tool that is used to control and manage data elements in a uniform manner and is therefore an aid to the DBA is the Dictionary/Directory System (DDS). The DDS can serve Database Administrators, system analysts, software designers, and programmers by providing a central repository for information about data resources across organization and application lines. It is a set of one or more databases containing data about an organization's information resources which can be retrieved and analyzed using standard database management system capabilities.(i.e., query languages, processors) [Ref. 7].

The range of information which can be held in a DDS is very large. The simplest system may only hold sufficient information to document, say, COBOL file structures. The more complex systems may hold design requirements, designed database structures, possible future structures, operational information, extensive details of access to data, and even network specifications. Such complex systems initially build around data structure recording facilities, but once developed, are really the data processing departments' own database [Ref. 8].

The primary advantages of a well designed DDS which are applicable to SPLICE are:

- It contains a unique identification, a set of physical characteristics, and a textual description for each of the data elements.
- It shows the relationships of elements to each other, and to components of the system.
- It specifies the source, location, usage and destination of the elements.
- It has validation and redundancy-checking capabilities.
- It contains security safeguards to control the accessibility to the data elements.
- It has a command language.
- It has reporting capabilities, such as:

  - predefined management-oriented, statistical or summary reports

  -ad-hoc user-defined reports

  -cross-reference reports

  -elements usage reports

  -audit trail reports

  -change-effect reports

  -error reports
- It has retrieval capabilities, such as keywording, indexing, and online queries.
- It has facilities for interacting with a DBMS [Ref. 5].

A DDS would be useful to SPLICE as a documentation tool. Within the applications environment, analysts and users of the DDS would be required to define system data definitions and records as well as other elements of the dictionary. In the process, old definitions would be updated, outdated definitions would be discarded and new definitions would be input. Redundancy-checking capabilities would prevent two

elements from having different descriptions. Use of a DDS would enforce establishment of standard data definitions and descriptions for those applications programs used by the SPLICE system.

When the DDS has been designed and is part of the SPLICE system, it could serve as an important basis for the future development of projects tied to UADPS-SP. By cataloging data requirements resulting from requirement analysis activities [Ref. 9], the DDS becomes an aid in the development of new programs. Eventually, the DDS for SPLICE could facilitate conversion of the present file-oriented application system to a DBMS-oriented system when the DBMS becomes available.

The key aspect of a DDS is that it provides resource independence. Resources are protected from changes in other resources. Modifications or changes may be made within the DDS and this would not necessitate changes in the applications programs. Conversely, changes in the applications programs would not necessitate modifications of the DDS data entities. In a distributed environment, this resource usage flexibility is particularly advantageous. Uniformity in resource description standards, enforced by a SPLICE DDS, would enable a smoother and more convenient interface of large application systems like APADE and UADPS-SP [Ref. 7].

B. PASSIVE AND ACTIVE DDS

DDSs can be grouped according to whether their dictionary/directory function is active or passive. Further, DDSs can be subdivided into free-standing (independent) or dependent depending on their implementation. Table I presents a schematic representation of this classification.

TABLE I

DDS Functional Classifications

SOFTWARE TOOL	FUNCTION	IMPLEMENTATION
DDS	ACTIVE	FREE-STANDING
		DEPENDENT
	PASSIVE	DEPENDENT

A passive DDS is a software package that does not require that the processes or system components depend on the DDS for their data. In its simplest form, a passive DDS only registers the data elements for programs and processes on an after-the-fact basis as a documentation facility. Passive DDSs usually are embedded functions within another system, serve as the data and file pre-definition mechanism, and are an integral physical part of another system. Since the passive DDSs are embedded within another system, they are necessarily oriented towards the characteristics and internal representation of that system.

An active DDS is a separate and distinct software package that functions mainly as a tool for identifying, locating, controlling, reporting, and manipulating the information about data elements in a database. It is a basic tool within the database environment that can assist the data administrator, the system analysis/designer, and

the programmers in managing, planning, and evaluating the collection, storage, and usage of the data resources. DDSs are said to be active with respect to a program or process if and only if that program or process is fully dependent upon the DDS for its data [Ref. 5].

The existence of active DDSs as separate entitites rather than as a part of another system is a recent innovation in the area of data management. They may be implemented in such a way that they require a DBMS to function consistently. A further subdivision of DDSs tends to make clear the implementation of these packages. They are free-standing (independent) or dependent. A point that needs to be made is that both of these divisions function principally as DDSs, and they are different only in their implementation.

The major differences between an active and a passive DDS are:

- The active DDS is a self-contained system, whereas the passive DDS is an internal function of another software system.
- The reporting and retrieval capabilities are extensive for the active DDS, and modest for the passive.
- Active DDSs have more extensive security control over the data elements.

1. Free-standing

Free-standing DDSs are self-contained, and perform the basic functions of controlling and managing the data elements without dependence on a DBMS. However, they may use programs not specifically written for the DDSs in order to enhance their capabilities and performance. A DDS that is free-standing may support one or more DBMSs through the use of interfaces, achieving mutual benefit from this association. It should be emphasized that the free-standing DDS

does not depend on the DBMS to function but the use of DBMS interfaces can provide the database administrator with a greater degree of control over the DBMS. It is even possible for free-standing DDSs to have interfaces to more than one DBMS simultaneously. Free-standing DDSs are also known as Generalized or Independent DDSs [Ref. 5].

2. Dependent

The dependent DDSs are separate software systems that are specifically tailored to a general purpose DBMS. They provide the DBMS with control and management of the data elements by supplying the DBMS with the descriptions, definitions, locations, and cross-references of the data elements. In turn, DBMS resources such as file structure and access methods are made available to the DDS. Because the dependent DDS is designed and implemented to be DBMS-specific, the portability of this type of DDS is restricted to installations having that particular DBMS.

C. SOFTWARE INTERFACES

The software interface provides a formatted pathway, enabling the DDS to provide a set of data attributes to other software systems such as compilers and Data Definition Language (DDL) processors and enabling these systems to retrieve and update information in the DDS either statically or dynamically [Ref. 6].

The static interface links the DDS with another system indirectly via the extraction of a file of formatted data. For example, the DBA enters into the DDS all pertinent transactions to describe the database. After reviewing the accuracy of this database description the DBA activates a DDS command, say GENERATE, that uses this description to provide a file containing DDL translates this generated DDL

into a schema file that the run-time unit of the DBMS can access.

Static interfaces differ depending upon whether they interface the DDS with user-written programs or with vendor-supplied software packages. Static interfaces for programs written in languages such as COBOL and PL/1 produce file, record, and database descriptions for the user programs from the DDS. These interfaces feature edit capabilities, format options, and various other functions to make the interface more flexible. Static interfaces for software packages such as DDL processors, communication monitors, and query processors, produce formatted statements for those packages or create specially encoded control files for their use. The options for customizing the interface with these types of software packages are more limited than are the options provided for interface with user-written programs. Also, these packages generally have more rigid language formats and the interface statements are usually used only by the DBA [Ref. 6].

Static interfaces are prevalent because of their utility, compatibility and efficiency. The static DDS can be made compatible with many versions of other software packages (i.e,several different DBMS's) and can be developed independently of the source code of particular software packages. A disadvantage to the user of a static interface is the extra effort that may be required to generate and catalog data for the DDS. Plus, the static interface itself has no capability of updating the data of the system with which it interfaces and therefore data in other systems can become inconsistent with data in the DDS.

Dynamic interfaces provide direct access of the DDS to other software modules. This direct access is commonly achieved by high-level interface commands that shield the software package from the physical details of the DDS.

Dynamic interfaces provide consistency control and capabilities for both update and retrieval. Changes to the DDS are automatically reflected in the next execution of any software packages to which the DDS is interfaced. A software package can directly retrieve and update data stored in the DDS. For example, a COBOL preprocessor can retrieve file descriptions from the DDS and update the DDS to reflect the new compilation date in the identification of invoking modules. All requests by software packages for data are routed through the DDS by a dynamic interface function, so the security and validity checks of the DDS are always applied [Ref. 6].

Use of a dynamic interface incurs significant overhead due to the size and complex structure of the DDS. Application development support aids, such as preprocessors, source program managers, and design aids, generally can afford this overhead because response time is not critical but on the other hand, efficiency is critical for transaction-processing systems (i.e. a query processor) that reference the DDS. To reduce the potential overhead, common queries may be precompiled and stored in the DDS or the software packages can retrieve all the data required for a transaction at once and therefore future accesses for this transaction become memory lookups.

D. CONVERT FUNCTIONS

The integration of a DDS into its environment is provided by convert functions. These functions alleviate some of the problems encountered in initially populating a DDS. The convert functions of a DDS scan source programs, database descriptions, and teleprocessing environment descriptions and automatically produce maintenance transactions. The DBA will still have to scrutinize these

generalized transactions for the possibility of redundant and nonstandard entries. An example would be if the naming conventions used in source programs are uncontrolled, then the DBA will find it necessary to change details in many of the transactions in order to impose standardization.

Convert functions are offered to convert data from both user-written programs and from a DBMS and its related components. This is shown in a DDS that furnishes conversion facilities for several programming languages (i.e.,COBOL, PL/1, assembler), as well as for the DDL processor and report writer of a given DBMS. Convert functions have four significant characteristics; 1) the content of the generated transactions, 2) the input file, 3) the command options, and 4) the source program analysis. The data dictionary maintenance transactions that are created by convert functions usually also contain the relationships between data entities (i.e., between a record and an element) and most language clauses. For example, element transactions generated by a COBOL convert function capture data for the DDS from the program's OCCURS, PICTURE, and USAGE clauses. A convert function is considered to be equivalent if the DDS can regenerate the input source data description from the DDS data. The ability to analyze the data of source programs can make the DDS a valuable tool for auditing adherence to software control techniques. The DDS can detect and disclose inconsistencies in the names, formats and clauses (comments, initial values) of the record definitions used in several programs and the corresponding standard record definitions in the DDS [Ref. 6].

E. ENVIRONMENTAL DEPENDENCY

The environmental dependency characteristic of a DDS is determined by its reliance on a specific hardware

configuration, an operating system, a DBMS or a teleprocessing monitor. The DDS should be able to operate in all environments with no loss of efficiency and functionality but at the present time this is not attainable [Ref. 6].

The amount of environmental dependency between a DDS and a DBMS is determined by the degree to which the DDS relies on the DBMS for data management services and the source of data used by the DBMS for access to stored databases. The degree of environmental dependency is broken into INDEPENDENT, DBMS-APPLICATION, and EMBEDDED approaches.

In the INDEPENDENT approach, the DDS is autonomous. The DBMS maintains its own source of data and the DDS uses none of the DBMS utilities for providing data dictionary management functions. The benefits of this approach are that it makes the dictionary more portable and will probably be possible to use the dictionary on a variety of hardware and operating system configurations.

In the DBMS-APPLICATION approach, the DDS appears to the DBMS as just another database. The DBMS maintains its own data for each database and these are separate from the DDS. The DBMS utilities provide most of the DDS management functions, but data is defined separately for the DBMS and for the DDS. The DDS will interface dynamically with only one DBMS and its related components, but there can be static interfaces to other DBMSs that operate with the same hardware.

In the EMBEDDED approach, the DDS is actually a component of the DBMS. This approach provides complete integration of the DDS. The DBMS utilities provide the DDS management facilities and the DBMS uses the DDS to direct access to stored databases. No other internal directories exist for the DBMS, and its related components rely completely on the DDS for data. For example a query processor extracts user views from the DDS and the DBMS

applies integrity constraints specified in the DDS before storing a data element.

The DBA needs to decide what functions this powerful tool in a database environment is to perform. The DBA will have to evaluate the pros and cons of each function that has been discussed and then weigh them according to the relative importance of each function to the DBA's environment. For example, does he want the software interfaces to link to other systems statically or dynamically, how much does he want the convert functions to do automatically, what price is he willing to pay for this service, and what reliance does he want the DDS to have on the hardware configuration, an operating system, a DBMS, or a teleprocessing monitor? Once these questions are answered the DBA must be sure that the functional requirements of his environment have been met.

F. CONTENTS AND LOGICAL STRUCTURE OF DDS

A useful DDS will contain more than merely information about data. It is desirable that the DDS represents such system resources as data, hardware, software, transactions, personnel, and documents. Entities, attributes, and relationships associated with these resources are presented below.

Data resource information should include the following entities: data elements, data groups, schemas/subschemas, records, files, and databases. Attributes for these entities must be determined by the anticipated usage of the DDS. [Ref. 6] suggests typical attributes for the data element entity and these appear in Table II. These attributes have been chosen from existing commercial data dictionary/directory systems. Attributes for the file entity have been suggested in the SPLICE specifications [Ref. 10], and appear in Table III.

TABLE II

Data Element Attributes (from Allen et al 1982)

Type	Language names
Range	Repetitions
Length	88 Levels
Unit of Measure	Key
Usage	Default value
	Display format

TABLE III

File Entity Attributes

File name	Format (seq, random, bin)
locations	Access control
Size (in bytes)	Access security protection

Relationships are associations between one or more data entities and are also a function of anticipated usage. Figure 2.1 is a possible network structure relating data entities. This structure, implemented in the appropriate DBMS would provide answers to the following queries via conventional query languages and processors:

- "List the record layout (all data elements and keys) for the Master Stock Item Reference (MSIR)?"
- "What locations in the network stock item 5905-00-255-3699, resistor, fixed, composition?"
- "What files and records contain the data element NATIONAL-STOCK-NUMBER?"
- "Who is the manufacturer of part-number 248W-24-A?"

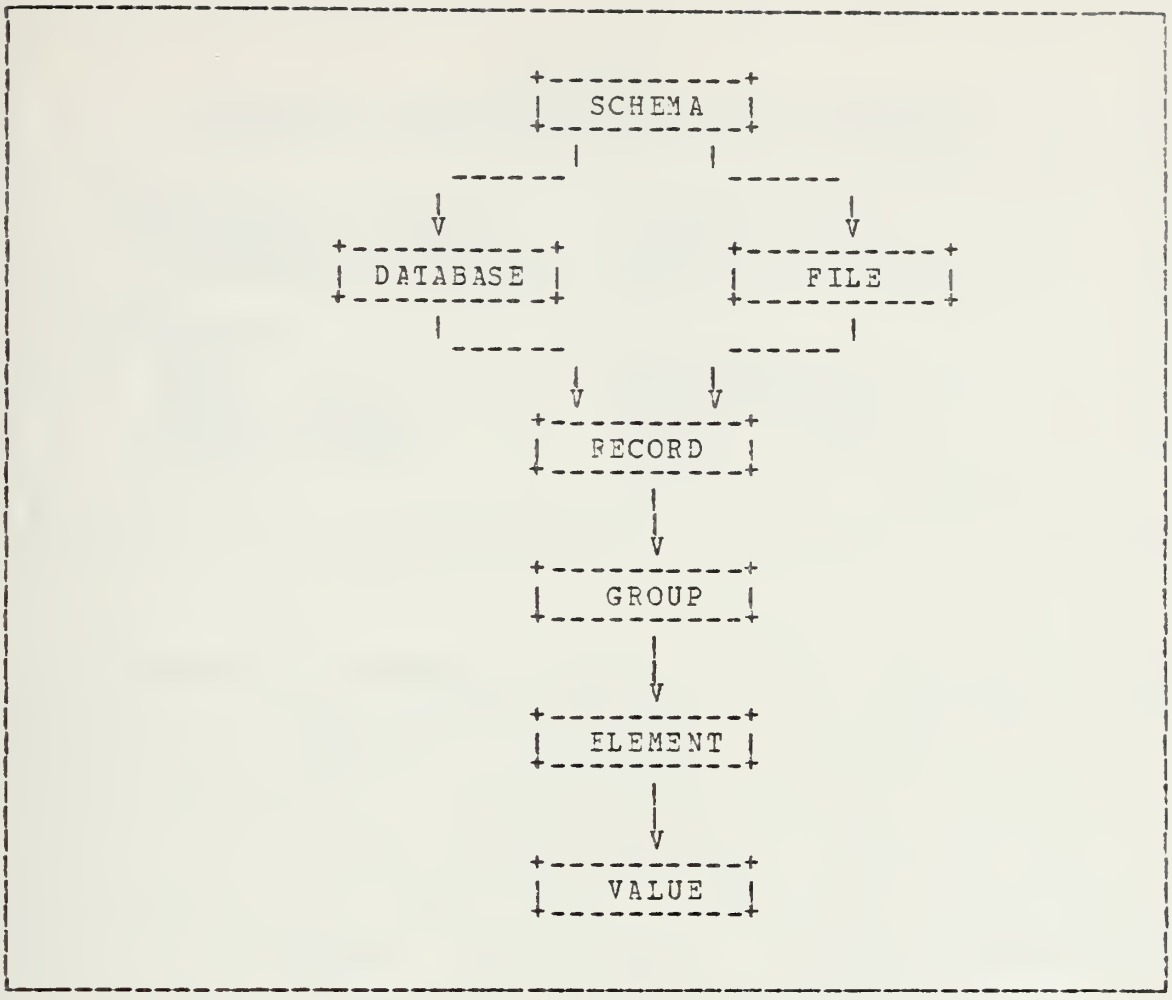


Figure 2.1 Logical Structure Describing Data Resources (from Schniedewind and Dolk 1983).

Hardware entities and attributes identified in [Ref. 10] have been specified as part of the Configuration Management System (CMS) for SPIICE. A selective sample is shown in Table IV. The DDS could be of value regarding hardware resources if it could display topological information about all or part of the SPLICE network. One logical structure which might provide this capability within a conventional DBMS environment is given in in Figure 2.2 [Ref. 7]. Possible queries might include:

- "How many terminals are located within each LAN?"
- "Which LANs have database processors?"

TABLE IV

Selected Hardware Entities and Attributes

Entities	
Processing system	Concentrators
Secondary storage	Terminals
Communications system	LAN I/O peripherals
Attributes	
Type	Features
Model	Description
Model number	Docu. references
Serial number	Usage by site
Mfger's number	Cost
Source	Maintenance activity

Recommended software entities and attributes from [Ref. 10] are summarized in Table V. One possible logical structure for software, transaction, and report entities is given in Figure 2.3. The advantage of this kind of structure is that one can extract data flow information from it relatively easily. Thus a query to the effect of "construct a data flow analysis (input files, modules/transactions, output files, reports) for the APADE system" should be reasonable to implement. Inclusion of these resources in the DDS contributes greatly to the viability of the DDS as a systems analysis and design tool.

The DDS can catalog documents other than reports. A summary of entities and attributes for various kinds of documents is shown in Table VI [Ref. 10].

Personnel is another resource which can be represented in a DDS. Information about users, account numbers, and access authority may have important impacts on security considerations. DBAs could find it useful to have access to data about programmers, analysts, and the programs/systems which they are responsible for. Because of the diversity of

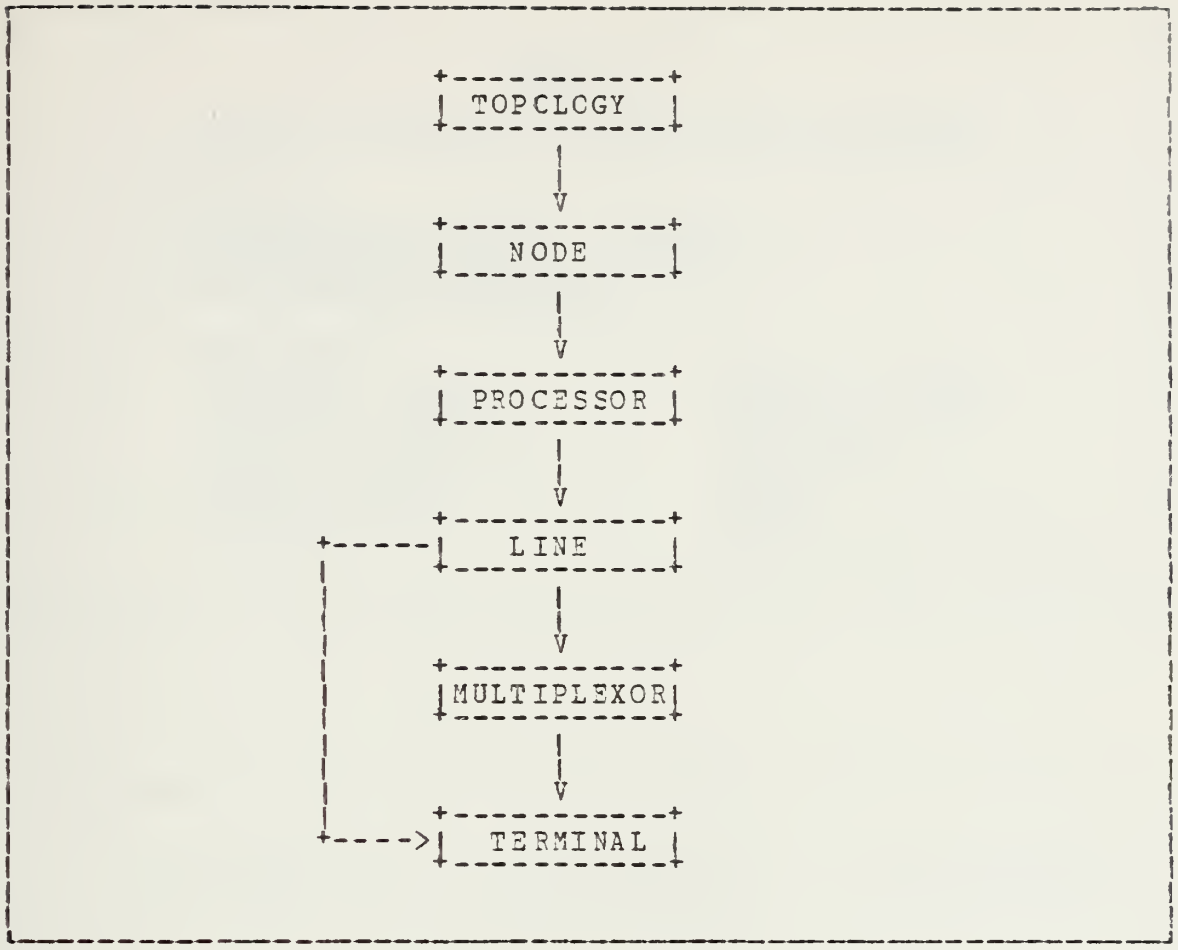


Figure 2.2 Logical Structure of Hardware Resources (from Allen et al 1982).

possible relationships between users and other resources in the DDS, no structural representation will be addressed. As with the other resources discussed, the applicable relationships will be a function of the intended usage of the DDS.

G. IMPACT OF A DDS

It has been stated in the introduction that there is increasing awareness of the importance of data as an organization resource and a recognition that it must be

TABLE V

Selected Software Entities and Attributes

Entities
- Operating system
- Operational support system
- Environmental system
- Application software

Attributes

Program-id	Date released
Revision number	Product number
Revision date	Source
Date compiled	Features
Type of compiler	Documentation
Patch level	Usage
Change level	Cost
License	Maintenance activity

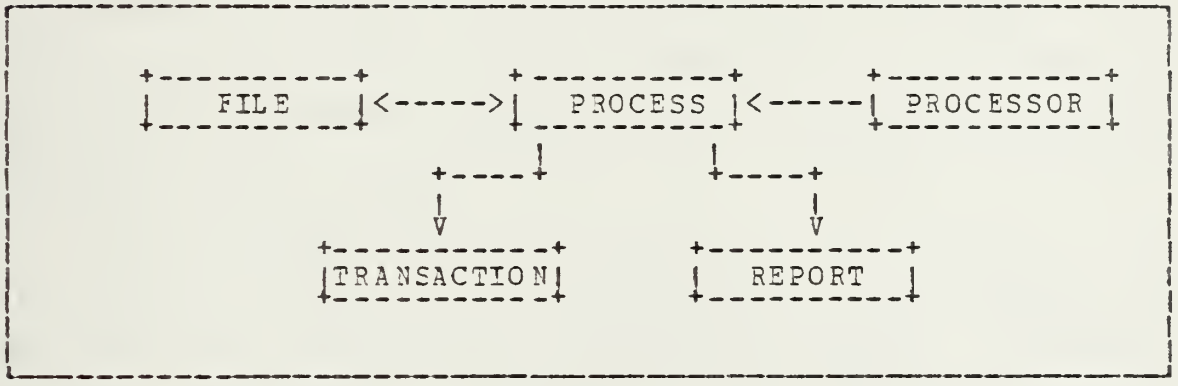


Figure 2.3 Logical Structure for Software, Transactions, and Report Resources.

controlled as carefully as other major resources within the organization. When management has gained control of the data resource by recording all information about it, it can then begin to impose controls over the availability of the resource and to develop standards for its use. In this way the data resource can be managed to the best advantage for

TABLE VI

Document/Report Attributes

Name	Source
Number	Feature
Product number	Description
Release date	Quantity
Revision number	Cost

the existing information system and can be effectively redeployed to meet changing information requirements. To obtain this control of the system resources, a DDS was described as being the software tool that will help to insure the best use of the system resources.

In the process of gaining control of the resource, individuals will lose the freedom to define and use resources in their own way to satisfy their needs alone. In return for this loss of freedom, the user will have access to accurate information about the definitions and usage of all the data contained in the system.

A DDS improves a Database Administrator's control over the design and the use of the database. A DDS enables him to control and document the formulation, meaning, and usage of data structures. It enables him to evaluate existing data redundancy, provide accurate data definitions for inclusion in programs, and it enables him to insure that management's requirements for data standards are obeyed. The most important benefits are that the DBA will be able to assess quickly the impact of any proposed change to commonly used data and to estimate the likely cost and time-scale of any such change. During this changeover, the DBA can be assured of completeness (because the DDS will show all the programs, files and reports affected) and accuracy (because the DDS can generate new coding to reflect the change).

The systems designer will be able to use the DDS as a central source of information. It will allow him to make use of data that is already available and help him to avoid duplicating data definitions, thereby preventing redundancy and inconsistency. The DDS can generate data files to be used for system testing, enable him to check the contents of data files produced during systems testing, and provide the designer with documentation of the system.

Application programming management will be able to enforce data definition standards by insisting that all data definition coding is produced by the DDS. This will also make it easier to control the implementation of design changes that arise during development. For the application programmers themselves, much of the tedium of writing large amounts of coding will be removed. As the DDS becomes more involved, greater amounts of coding will be generated by the DDS itself. Further, assistance with the generation of test data and the checking of results by the DDS will reduce application development time and improve the accuracy of the finished programs. Once management has used the DDS to assess the impact of a proposed change, the task of amending the applications can proceed with increased confidence. The DDS's cross-referencing will show the maintenance programmer precisely what programs will be affected by a particular change. As programs are constantly amended, the DDS can be used to record the changes ensuring programming management control of the progress and completeness of the change. A big impact that the DDS has on the current system is that before allowing the changed programs to replace the production versions, the DDS can be used to generate test data and check results.

The operations department will be impacted since it will be able to maintain the privacy of data by reference to the DDS to check who is allowed to use what data. The DDS will

aid the department in the creation of job control language parameters, control of different versions of program libraries and data files, distribution of multiple copies of output, and discovering the source of invalid data.

In summary, it becomes apparent that the introduction of a DDS will have an effect at many levels in an organization. It will create new tasks, but in return will reduce the effort required by such activities as documentation, coding of programs, creation of test data files, checking and auditing of output files. The DDS will enable management to control data processing at all levels and will provide an effective means of communicating data processing requirements between user departments, operations departments, and the ADP department. System resources will be recognized as immensely important corporate resources and will be managed and controlled accordingly.

III. TANDEM DDL DATA DICTIONARY AND DISTRIBUTION STRATEGIES

A. DICTIONARY

The need to manage data resources requires systems that identify, describe, define, and relate the basic unit of information, the data element. The TANDEM Computer Company offers a Distributed Database Management System (DDBMS) called ENCOMPASS that handles this need. ENCOMPASS consists of standard software products: DATA DEFINITION LANGUAGE, ENFORM QUERY LANGUAGE/REPORT FORMATTER, TRANSACTION MONITORING FACILITY, PATHWAY TRANSACTION PROCESSING SYSTEM, and ENABLE SCREEN COBOL GENERATOR [Ref. 11,12,13]. The combination of these software tools works with the TANDEM NONSTOP systems to provide a reliable and high performance system that is capable of accommodating peak workloads, hardware expansion, database modifications, and application growth.

The Data Definition Language (DDL) is a language processor that enables the user to design and create a data dictionary. This creation takes place through a combination of statements and commands that can be used both interactively and in batch mode. By using the DDL statements, the user can define or modify the structure of the database, generate file utility statements to create database files, and provide source language data definitions output for COBOL, FORTRAN, and TAL compilers [Ref. 13]. This interaction is illustrated in figure 3.1.

The actual DDL looks like COBOL and provides two types of statements: DEFINITION statements and RECORD statements. The DEFINITION statement is a data description independent of any record that specifies fields and groups. The RECORD

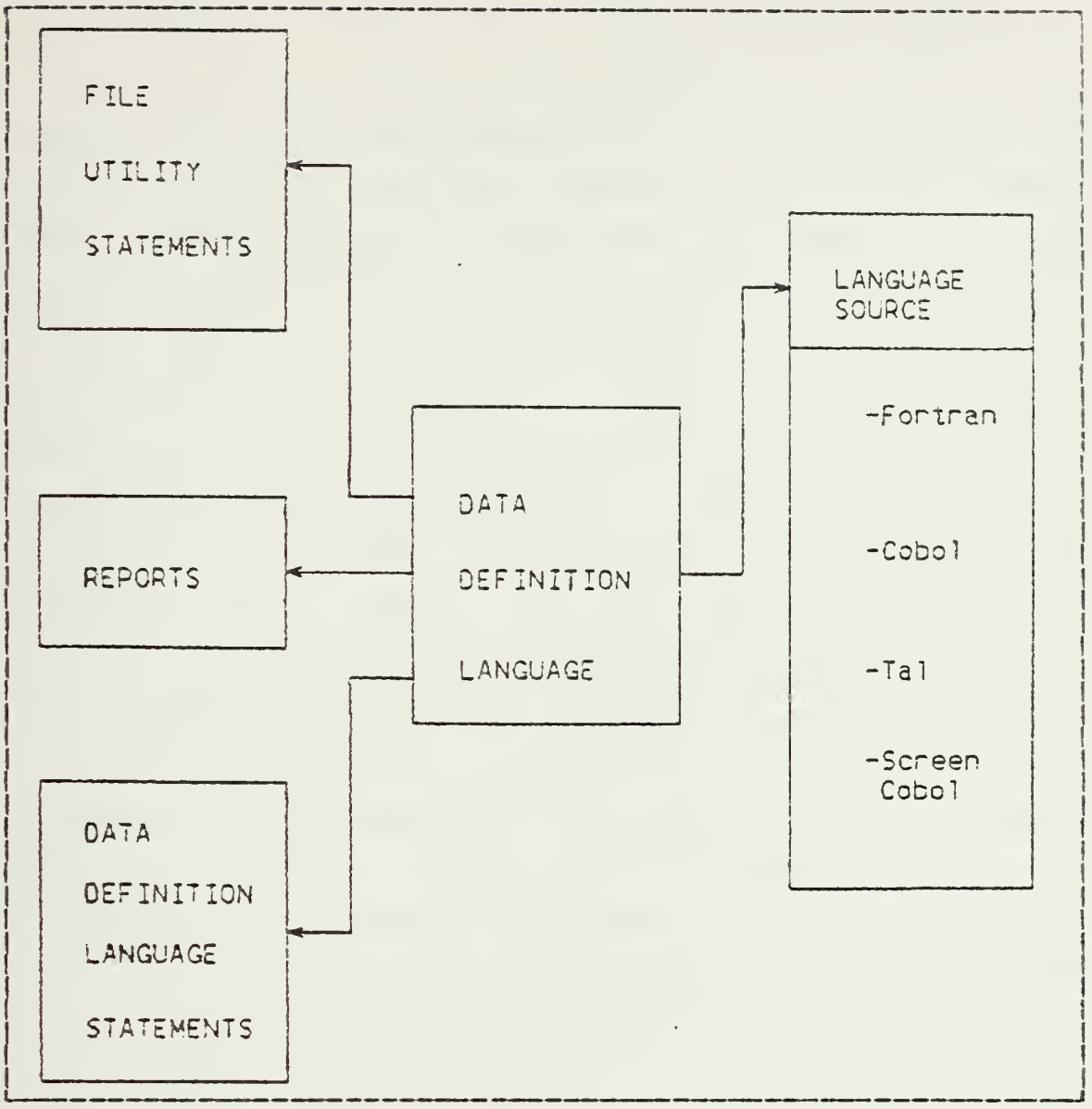


Figure 3.1 Interaction of the DDL.

statement defines record structure and file attributes, and can refer to a DEFINITION. When this happens the groups and fields of the DEFINITION are inserted into the RECORD statement at the point where the DEFINITION name is written. Many RECORD statements can share the data structure of the same DEFINITION. This application of DEFINITION statements

makes it easy to standardize a data structure throughout a large database. This insures consistency of nomenclature and data types among different records, files, and programs. The DDL will take every DEFINITION and RECORD statement in the DDL input file, plus user comments, and catalog them to form a data dictionary. The data dictionary will be a complete, centralized description of the data structures, records, and files that make up the database.

This data dictionary will essentially be a database about the system databases. It will provide a powerful development tool, a central source for data definitions, and complete database documentation. There are two main functions of the data dictionary in the realm of the ENCOMPASS DDBMS. First, the DDL data dictionary will provide a database of what kinds of information are maintained, such as stock numbers, document numbers, or routing identifiers. Secondly, it supports ENFORM by providing "mapping" information to describe the file type and access information of the database. The query processor of ENFORM uses this mapping information to determine the most efficient way to access the database.

A feature of the data dictionary, once it has been compiled by the DDL, is that it becomes a single source for all data definitions in the database. This means that every program that accesses a database can use data declaration source generated from the dictionary. Therefore, the DBA can exercise control over the structure of the database. The data dictionary also stores information in a form accessible to the ENFORM QUERY/REPORT WRITER, so users can easily prepare ad hoc dictionary reports. One of the best features of the TANDEM data dictionary is that separate data dictionaries can be stored on separate disk volumes for different databases. This feature allows concurrent users of different dictionaries to perform database interactions

without access conflicts that can cause bottlenecks [Ref. 13].

Since we now know what the data dictionary within the ENCOMPASS DDBMS is made up of, what it provides, what it produces, and what its features are, we must now ask the question what can this data dictionary do for SPLICE? The make-up of the TANDEM dictionary will be evaluated according to the basic profile given in the earlier part of this work.

B. DATA DICTIONARY COMPARISONS

The following comparison of the below described data dictionaries is intended to give a rough evaluation of the TANDEM product relative to what is currently available from other vendors. In no way are we endorsing a particular product. The intent of this limited survey is to look at selected features offered by the different software packages and note how the four products compare without making quantifiable judgements whether one or another is good, bad, superior or inferior. The information provided in the following tables was derived from [Ref. 5,6,11,12,13,14]. The section begins with a brief general description of each of the four data dictionary packages considered.

1. Data Control System (DCS)

The Data Control System (DCS) is a dictionary system produced and developed by Cincom Systems Inc. The system is implemented as an application program that requires Cincom's DBMS which is called TOTAL. DCS was previously referred to as the Cincom Data Dictionary.

2. Datamanager

Datamanager is a free-standing data dictionary system that is developed and marketed by MSP Inc.

Datamanager is the nucleus of a family of products. It contains the facilities for creating and maintaining dictionaries used in a traditional file environment. Datamanager also has dictionary query facilities as well as a report generator. Datamanager is not tied to a particular DBMS and is capable of interfacing to IDMS, ADBAS, IMS, TOTAL and SYSTEM 2000.

3. DB/DC Data Dictionary

There are two versions of the DB/DC Data Dictionary which is a product of International Business Machines (IBM) Corporation. One version is for operation under OS/VS and the other is for operation under DOS/VS. The DB/DC Data Dictionary is a DBMS-application program with several management features which have evolved since the product was first released in 1974.

4. DDL Data Dictionary

The Data Description Language (DDL) Data Dictionary produced by TANDEM Software Products Inc. is a DBMS-application approach that requires the TANDEM DBMS called ENCOMPASS. The TANDEM format for dictionary file entries was used in the preliminary DDS design proposed in this thesis.

TABLE VII

Data Dictionary Comparisons

Software Package Name	DATA CONTROL SYSTEM	DATAMANAGER	DB/DC DATA DICTIONARY SYSTEMS	DDL DATA DICTIONARY
Vendor	Cincom Systems Inc.	MSP Inc.	IBM	TANDEM Computers Inc.
Operational Mode	Batch	Batch and online	Batch and online	Batch and online
Use of DBMS	Requires TOTAL	Independent	Requires IMS/VS or DL1 DOS/VS	Requires ENCOMPASS
Online Queries?	No	Yes	Yes	Yes
Online Query Language?	Not applicable	Free-form	Fixed-form	Free-form
Synonym?	Yes	Yes	Yes	Yes
Keyword Capability?	Yes	Yes	Yes	Yes

TABLE VIII

Data Dictionary Comparisons (cont.)

Software Package Name	DATA CONTROL SYSTEM	DATAMANAGER	DB/DC DATA DICTIONARY SYSTEMS	DDL DATA DICTIONARY
Data Structure	Network	Hierarchical	Hierarchical	Relational
Validation	D/D facility	D/D facility	D/D (limited) and user written	D/D Facility
Redundancy/ Inconsistency Check?	Yes, DBMS facility	Yes, D/D facility	Yes, D/D facility	Yes, DBMS facility
Definition/ Description?	Yes, narrative description	Yes, narrative description	Yes, narrative description	Yes, narrative description
Relationships?	Yes	Yes	Yes	Yes
Owner/ User?	Yes	Yes	Yes	Yes
Ad Hoc	Yes	Yes	Yes	Yes

C. AN EXPLANATION OF THE TERMS USED IN THE SURVEY

SOFTWARE PACKAGE NAME: Name or acronym by which the D/D is known.

VENDOR: Name of the company that markets the product.

OPERATIONAL MODE: Mode in which the D/D operates, batch or online.

USE OF A DATABASE MANAGEMENT SYSTEM (DBMS): Note inferences to, or requirements for, a DBMS.

ONLINE QUERIES?: Does the D/D have online query capability?

ONLINE QUERY LANGUAGE?: Does the query language require fixed-form or free-form queries?

SYNONYM?: Can a synonym (word or abbreviation) be used as a substitute for the data element identifier?

KEYWORD CAPABILITY?: Can an element be identified as a keyword?

DATA STRUCTURE: What data structure is supported by the D/D, network, hierarchical, or relational?

VALIDATION: Is data validation performed by the D/D or by some other means?

REDUNDANCY/INCONSISTENCY CHECK?: Is there a specific feature that identifies whether the data element is redundant or inconsistent?

DEFINITION/DESCRIPTION?: Is there a facility for defining and/or describing a data element in narrative form?

RELATIONSHIPS?: Is there a capacity for specifying the relationship of a data element to another data element or to a higher level structure?

OWNER/USER?: Is there a facility for indicating authorized users (persons or programs) or owners of data elements?

AD HOC: Can the user structure his own reports, on an ad hoc basis?

D. ACTIVE OR PASSIVE

The TANDEM data dictionary function within the SPLICE project would have to be categorized as an active/dependent DDS. The reason for this classification is that the software package does require the Navy Supply System components to depend on the DDS for their data, and is a software package that functions mainly as a tool for manipulating information about data elements in a database. The dependent classification is attached since this software tool is specifically tailored to the ENCOMPASS DDBMS. In this format the DDS will provide the DBMS with control and management of the data elements and in return the DBMS resources, such as file structure and access methods, will be made available to the DDS. Since the DDS is designed and implemented within a specific DBMS environment its portability will be restricted to installations having this DBMS.

E. SOFTWARE INTERFACES

The software interface provided in the ENCOMPASS DDBMS is a static interface that links the data dictionary with other systems indirectly via the extraction of a file of formatted data. The dictionary contains the description of the records within each file and identifies primary and alternate keys. The relational QUERY/REPORT writing language ENFORM uses these as a template for accessing the records. This accessed information will provide "mapping"

information to describe the file types and to describe the database. The query processor of ENFORM then uses this information to determine the most efficient way to access the database. By supplying the software package to interface with the data dictionary the user has acquired edit capabilities, format options, and various other functions to make the interface more flexible.

The drawbacks in having static interfaces and using vendor-supplied software packages compared to the option of using dynamic interfaces come in several areas. Changes to the data in the data dictionary will not mean that the data of the system with which it interfaces will be automatically updated and therefore data in other systems can become inconsistent with data in the data dictionary. Software packages cannot directly retrieve and update data stored in the data dictionary nor are all requests by these packages routed through the data dictionary so that security and validity checks of the data dictionary can be applied. The use of static interfaces is prevalent in the world of dictionaries because of their utility compatibility, efficiency, and small overhead. Since SPLICE is largely concerned with efficiency in performing transaction-processing, this static interface approach will be a plus for the project.

F. CONVERT FUNCTIONS

The convert functions offered by the TANDEM software packages do not completely match the definition given by [Ref. 6]. Convert functions are described first as being able to scan source programs, database descriptions, and teleprocessing environment descriptions and automatically produce maintenance transactions that will spare the DBA many hours of manual effort. Secondly, the convert

functions are offered to convert data from both user-written programs and from a DBMS and its related components. The TANDEM DDL does not perform the first task but in the area of the second function it does operate very strongly as a conversion facility for programming languages COBOL, FORTRAN, and TAL, as well as for the DDL compiler and the report writer of ENFORM. The DDL compiler, once it has been given a list of DDL statements, can produce the following files:

- a data dictionary
- a file utility program for file creation command source
- a data declaration source for COBOL, FORTRAN, TAL
- a schema report summarizing each record's structure and each file's access keys

The ENSCRIBE file utility program (FUP) commands will be used to create the actual database files with the specified attributes. The data declaration source generated by the DDL is available either when the schema is first compiled or by accessing the data dictionary at any other time. The DDL will also provide edit checking for all three source outputs. For example, if the user programmer asks for COBOL source output, the DDL checks to make sure that the schema includes no COBOL reserved words, and will report what items are unsuitable for the selected language. [Ref. 13].

The question is what does all this mean for SPLICE? By gathering all of the various database definitions and data structure declaration functions into a single language, the DBA gains control over the database design and implementation. It also gives the DBA a data dictionary that documents the database, and can be used in the ongoing process of database management. With the data dictionary being able to regenerate the input source data description from the dictionary data, a valuable tool for auditing adherence to software control techniques is gained. This means the data

dictionary within the SPLICE environment will be able to detect and disclose inconsistencies in the names, formats, and clauses of the record definitions used in several programs and the corresponding standard RECORD definitions in the data dictionary [Ref. 6].

G. ENVIRONMENTAL DEPENDENCY

There are three levels of environmental dependency between a data dictionary and a DBMS: independent, DBMS-application, and embedded. The environmental dependency characteristic of a data dictionary is determined by the degree which a data dictionary relies on the DBMS for data management services and the source of data used by the DBMS for access to stored databases [Ref. 6].

The data dictionary within the TANDEM schema would be classified as having DBMS-application characteristics. This allows random processing, on-line updates, query processors, and report generators to be more efficient. The DBMS-application method used by TANDEM means that "ad hoc" queries and special reports can be processed quickly and that data redundancy can be avoided. It also means that dictionary data descriptions can be used directly by the DBMS rather than having the dictionary generate DBMS data descriptions. The best advantage of the DBMS-application approach is that a customized data dictionary can be designed and implemented which meets the special needs of the SPLICE information resource environment [Ref. 7]. The drawbacks of this technique are that 1) it makes the dictionary less portable since it is tied to a specific DBMS and 2) it forces the user to buy and support this particular DBMS software. Applying this to the SPLICE project, these drawbacks are negligible since the project will be using the ENCOMPASS DBMS provided by TANDEM.

B. DISTRIBUTION CONSIDERATIONS FOR THE DATA DICTIONARY

In a distributed database environment there are a number of options for distribution of the data dictionary. Within the SPLICE system every lan will have a database management module for centralized control over the data. There are no distribution of databases within a lan [Ref. 4]. The data dictionary in the distributed environment itself becomes a distributed database. Databases may be distributed over the entire SPLICE network and the database functions are centralized within each lan. The goal here is to satisfy control and integrity of the data within the network.

I. DATA DISTRIBUTION STRATEGIES

The system architecture and the network database management system software determine the distribution strategies which may be considered. There are four types of distribution strategies for consideration.

1. Centralized-a single copy of the database is located at one node.

2. Partitioned-a single copy of the database is comprised of disjoint subsets or partitions located at various nodes.

3. Replicated-there are multiple copies of the database with each node having a complete copy.

4. Hybrid-there are multiple copies of subsets of the database where each node may have as many of the partitions of the database as necessary.

There are different advantages and disadvantages to each of the four distribution strategies mentioned. The primary considerations for discussion are reliability, data storage, retrieval and update response times and various control

mechanisms and tradeoffs in either software or communication costs.

1. Centralized

The centralized database concept has simplicity as its major advantage. Control and update problems are minimized since all the data are located at a single node. Drawbacks of this strategy include potential problems of reliability or availability. If the central node fails, the system is down completely. If there is a high volume of communications with the central node, there may be bottlenecks which slow down system response time. Depending upon the amount of secondary storage, there might also be a limitation on the possible size of the database. These considerations are applicable to each lan but are not of concern for the SPLICE system as a whole.

2. Partitioned

With the partitioned database distribution strategy there is still only one copy of the database however, it is divided into disjoint subsets or partitions which are assigned to particular nodes in the system. With this approach, the size of the database is not limited to the amount of secondary storage at a central node but is the sum of the total storage available within all nodes of the system. Retrievals and updates present less of a problem since they can be directed to the particular node where the data are located. Access to the local database partitions lessens the possibility of bottlenecks and may reduce communications costs. If the requirements are for data located outside of the local node, communications costs are greater and the delay in response time also increases.

The benefits of the partitioned approach are highly dependent upon the location of the data required to satisfy

user requests. This is also known as locality of reference. There is a high degree of locality of reference if the data is partitioned such that the data at a particular node is used almost exclusively by users at that node. If this is not optimized, database availability is limited in the system. With the partitioned approach, reliability of the system is improved over the centralized approach because failure at one node only causes a performance degradation, not failure of the whole system. In contrast, the availability of the entire database is less likely because failure of any node in the system is more probable than failure of the central node in the system. The partitioned strategy is most appropriate in a distributed environment where there is a limitation on secondary storage at the central node, where reliability of the system must be improved, or where there is a possibility for operating efficiencies to to be gained. A high degree of locality of reference in database access patterns implies that operating efficiencies can be gained through partitioning of the database. Within the SPLICE environment, determining the optimal partitioning strategy to ensure system efficiency poses a difficult problem.

3. Replicated

In the replicated distribution strategy each node within the system is allocated a complete copy of the database. The network database management system is responsible for coordinating the multiple copies of the database but there is not a problem of determining which nodes have which part of the database as there is with the partitioned approach. The replicated approach is most advantageous in the areas of reliability, availability and retrieval response time. As with the centralized approach, the size of the database may be limited by the size of the secondary

storage at each node. Faster response times for user requests are possible because there is no need to go outside of a particular node for database access. This also implies that communication costs would be cheaper as most network communication would be localized. With the replicated approach, reliability is high in terms of data availability and if a particular node fails, there is no lost portion of the database. Another copy of the complete database could be generated from a neighboring node. This simplicity of backup and recovery operations is another of the advantages of the replicated approach. If a node within the system fails, there must be controls in the updates that may be performed to maintain the integrity of the database. This is to say that independent updates of separate nodes must not be allowed or there will be likely data inconsistencies. Locking mechanisms must be employed to ensure data integrity.

The replicated distribution strategy is most appropriate where the database is not too large, reliability is critical to the system and inefficiencies of updates can be accepted. This implies a retrieval-intensive database system. The amount of overhead for communications and processors needed because of synchronization and control complexities in a replicated system is dependent upon the level of consistency required. Within the SPLICE system the data dictionary should be fairly static once implemented. With only limited update requirements, the replicated strategy for the SPLICE data dictionary is the recommended approach.

4. Hybrid

The hybrid distribution strategy is a combination of the partitioned approach and the replicated approach to data distribution. The database is partitioned into disjoint

subsets however, in this strategy there may be several replicated partitions. Each part of the database can be replicated any number of times and each node may have that portion of the entire database which optimizes performance at that node. This strategy, if properly implemented with a high degree of locality of reference, should limit the need for node to node communications thus reducing costs and potential communications bottlenecks. The hybrid strategy improves the reliability of the system over the strictly partitioned approach. The key advantage of the hybrid approach is flexibility in how the data are stored. Nodes with limited secondary storage may tailor their partition for functional optimization whereas nodes which demand a high degree of reliability and faster response times may duplicate a larger portion of the database to satisfy their needs.

The flexibility provided by the hybrid approach is achieved at the cost of system software complexity. The network DBMS must not only keep track of where data exist in the network but it must also be capable of synchronization of the data updates. Query optimization and query processing are nontrivial tasks in the hybrid environment. Because of the software complexity necessary to implement the hybrid strategy, the question is whether the flexibility to be gained by this approach is worth the tradeoff. A likely candidate for the hybrid approach might be a system with a large database that has only a few nodes capable of handling a replicated approach and where high reliability is essential in the nodes which have the secondary storage capacity to accommodate a large portion of the database.

J. EXAMPLE OF TANDEM DATA DICTIONARY FILES

The following schema consists of 20 objects (18 definitions and two records). These objects will be used to construct an example for the SPLICE dictionary and its related files. [Ref. 13] was used as a format guide and for explaination of the files which follow.

```
?COMMENTS


* Document Number uniquely identifies requestor, date and
* local serial number.

DEF    DOCNUM                HEADING "Document Number"
       02  Service           PIC A.
       02  Unit Id Code      PIC 9(5).
       02  Julian date       PIC 9(4).
       02  Serial num        PIC 9(4).

* National Stock Numbers are classified by groups
* catgorizing items by individual characteristics, and
* uniquely indentified within the Federal Supply System.

DEF    NSN                   HEADING "Nat'l Stock Number".
       02  Fed Sup Class     PIC 9(4).
       02  NIIN              TYPE *.

* Document Identifier which indicates the purpose and use of
* the document (i.e., requisition, referral, follow-up,
* status, etc.). The Document Identifier is a mandatory
* entry on each milstrip document.

DEF    DOCID       PIC AXX   HEADING "DOCUMENT IDENTIFIER"

* Routing identifier is used to represent the address of the
* intended recipient of the document; to denote the actual
* consignor of material; or to identify the supply activity
* originating the action.

DEF    ROUTID         PIC AX9    HEADING "ROUTING IDENTIFIER"

* Forecast level of expected demands for next quarter
* according to demand for the past six months.

DEF    DMDFORECAST    PIC 9(4)   HEADING "DEMAND/FORECAST"

* Minium items on hand to meet war reserve requirements.

DEF    RESERVATION    PIC 9999   HEADING "RESERVATION"

* A stocking level that is computed by adding demand during
* lead time plus safety stock.
```

```
DEF   RECRDERPOINT  PIC 9(4)    HEADING "REORDER/POINT"

* A managerial technique used to protect against a stockout.
* An order can be computed and placed so that the delivery
* will arrive when a certain level of inventory is still
* there.

DEF   SAFETY LEVEL  PIC 9(4)    HEADING "SAFETY/LEVEL"

* Number of units that have outstanding requisitions against
* them.

DEF   BACKORDER     PIC 9(5)    HEADING "BACKORDER"

* This gives the general category of a part, i.e., resistor.

DEF   PART NAME     PIC A(10)   HEADING "PART/NAME"

* The Media Status Code indicates the recipient of status
* and the means of transmission.

DEF   MEDIA STATUS  PIC X       HEADING "MEDIA/STATUS"

* This two digit alpha/numeric figure identifies who has
* inventory and technical responsibility for an item.

DEF   ACCOUNT/COG   PIC XA      HEADING "ACCT/COG"

* Priority combines the assigned force/activity designator
* and the appropriate urgency of need designator and enables
* the requisitioner to determine the appropriate priority
* code.

DEF   PRI           PIC 99      HEADING "PRIORITY CODE"

* Expected means of transporting items, and received in
* shipment status documents.

DEF   MODE          PIC X       HEADING "MODE OF SHIPMENT"

* National Item Identification Code uniquely identifies a
* line item within the Federal Supply System.

DEF   NIIN          PIC 9(9)    HEADING "NIIN"

* A description of the types of units under which material
* is issued.

DEF   UNIT OF ISSUE PIC AA      HEADING "UI"

* Cost of an item per unit of issue

DEF   UNIT PRICE    PIC 9(6)V99 HEADING "UP"

* Number of items of the object per unit of issue that are
* physically located at that particular stock point.

DEF   QTY ON HAND   PIC 99999   HEADING "ON/HAND"
```

```
* PRIMARY-KEY = DOCNUM
* ALTERNATE KEY = NSN

RECORD  DHF   FILE IS "$DATA.SUPPLY.DHF"    KEY-SEQUENCED

     02  DIC              TYPE DOCID.
     02  NSN              TYPE *.
     02  ACCT/COG         TYPE *.
     02  DOCNUM           TYPE *.
     02  RIC              TYPE *.
     02  PRI              TYPE *.
     02  QTY ON HAND      PIC 9(4)        HEADING "ON/HAND".

* KEY IS DHF.DOCNUM.
* KEY "NS" IS DHF.NSN.

END


* PRIMARY-KEY = NSN
* ALTERNATE KEY = PART NAME

RECORD  MSIR.  FILE IS "$DATA.SUPPLY.MSIR"   KEY-SEQUENCED

     02  NIIN              TYPE *.
     02  PART NAME         TYPE *.
     02  PURPOSE CODE      PIC A          HEADING "PURPOSE CODE"
     02  DATE ACTION       PIC 9(4)       HEADING "DATE LAST ACT"
     02  DATE INV          PIC 9(4)       HEADING "DATE LAST INV"
     02  UNIT PRICE        TYPE *.        DISPLAY "M<zzz,zz9.99>".
     02  QTY ON HAND       TYPE *.
     02  REORDER POINT          TYPE *.
     02  UNIT OF ISSUE          TYPE *.
     02  RESERVATION            TYPE *.
     02  BACKORDER              TYPE *.
     02  DEMAND FORECAST        TYPE *.
     02  SAFETY LEVEL           TYPE *.
     02  LEAD TIME              TYPE *.

KEY IS MSIR.NSN.
KEY "PN" IS MSIR.PARTNAME.

END
```

The Dictionary Definition File (DDF) is an unstructured file containing only one record. The fields that have any meaning are NEXT-OBJECT and NEXT-TEXT-ID. There are 5 other fields that give DDL version information. The NEXT-OBJECT is a counter the DDL compiler uses to assign object numbers to objects (the schema previously given has 20 objects-- 18 definitions and 2 records) as they are entered into the dictionary. This field will be incremented by one each time an object is added and this new value is assigned as the object number of the new object.

The NEXT-TEXT-ID is used by the DDL compiler in the same manner as the NEXT-OBJECT. Dictionary comments, PIC

strings, HEADINGS strings, DISPLAY strings, and VALUE literals are all text items. The previous schema has 75 text items and therefore the text count stands at 75. Figure 3.2, shows how this file would look.

NEXT OBJECT	NEXT TEXT-ID	DICTIONARY VERSION
20	75	1

Figure 3.2 Dictionary Definition File.

The Object Definition File (ODF) is a key-sequenced file that contains one record for every object entered into the dictionary. Figure 3.3 shows this file according to the schema previously given. The ODF file uses the object number from the DDF to identify the object. The object number is followed by object name with an object type (either ID the symbol for definitions or RD the symbol for records). The COMMENTS- TEXT- ID field is used for dictionary comments associated with the entire RECORDS (comment lines that immediately precede a RECORD statement in the DDL source schema). The values are assigned by the DDF.

The Object Build list (OBL) is a key-sequenced file that contains one record for each element of each object in the DDL schema. The records are identified by the object number that they received in the ODF. An element number is assigned that identifies each element within an object. A complete description of an element can be obtained from the OBL record (i.e., the elements name, data type, size, offset

OBJECT NUMBER	OBJECT NAME	OBJECT TYPE	COMMENTS TEXT-ID
1	DOCNUM	ID	
2	NSN	ID	
3	DOCID	ID	
4	ROUTID	ID	
5	DEMAND FORECAST	ID	
6	RESERVATION	ID	
7	REORDER POINT	ID	
8	SAFETY LEVEL	ID	
9	BACKORDER	ID	
10	LEADTIME	ID	
11	MEDIA STATUS	ID	
12	FUND CODE	ID	
13	PRIORITY	ID	
14	MODE	ID	
15	NIIN	ID	
16	UNIT OF ISSUE	ID	
17	UNIT PRICE	ID	
18	ON-HAND	ID	
19	DEMAND HISTORY	RD	61
20	MASTER STOCK	RD	64

Figure 3.3 Object Definition File.

within the object, and text-id number). Figure 3.4 shows how the OBI element records are linked to the ODF for a typical object, the DEMAND HISTORY FILE record.

OE NO	ELE NO	LV	OFF SET	ELEM NAME	GRP ELE	SZ	SRC DEF	COMNT TX-ID	HDING TX-ID	DSPLY TX-ID	PIC TX-ID	VAL TX-
1	0			DOCNUM	GRP	14		1	2		3	
2	0			NSN	GRP	13		7	8		9	
"												
"												
"												
20	0			DHF	GRP	41						
20	1			DIC	ELM	3	3	10	11		12	
20	2			NSN	ELM	13	2	7	8		9	
20	3			ACT/CCG	ELM	2	12	37	38		39	
20	4			DOCNUM	ELM	14	1	1	2		3	
20	5			RIC	ELM	3	4	13	14		15	
20	6			PRI	ELM	2	13	40	41		42	
20	7			QTY	ELM	4			61		62	

Figure 3.4 Object Build List.

The Object Text File (OTF) is a key-sequenced file that contains one record for each line of input of a dictionary COMMENT, HEADING string, DISPLAY string, PICTURE string, and VALUE literal in the DDL schema. The OBL links to the OTF on TEXT-ID fields, one to many. Figure 3.5 shows a partial layout from the previous schema.

TEXT NUMBER	LINE NUMBER	TX TP	TEXT LINE
1	0	C	DOCNUM IDENTIFIES REQS
1	1	C	DATE AND LOCAL SERIAL NUM
2	0	H	DOCUMENT NUMBER
3	0	P	A
4	0	P	9(5)
5	0	P	9(4)
6	0	P	9(4)
7	0	C	NATIONAL STOCK NUMBERS
7	1	C	ARE CLASSIFIED BY A 13
7	2	C	DIGIT NUMBER---------
"	"	"	"
"	"	"	"
67	0	H	UNIT/PRICE
68	0	D	M<zzz,zz9.99>

Figure 3.5 Object Text File.

The Record Definition File (RDF) is key-sequenced and contains one record for each RECORD in the DDL schema. The object number assigned to the RECORD through the ODF uniquely identifies each record. The RDF record also contains a DEF-NUMBER and if the record has been defined with a DEFINITION IS clause then DEF-NUMBER will hold the object number of the DEFINITION. If this is not true, then the DEF-NUMBER assumes the object number of the RECORD itself. The file type and file duration fields give details to the structure of the RECORD (i.e., unstructured, relative, entry-sequenced, key sequenced) and the permanence of the RECORD. The RDF links back to the ODF one to one plus

the RDF links to the OBL one to many. Figure 3.6, is an example of this file.

RECORD NUMBER	DEF NUMBER	TANDEM FILE NAME	FILE TYPE	FILE DUR	REC LTH
19	19	$DATA.SUPPLY.DHF	3	0	41
20	20	$DATA.SUPPLY.MSIR	3	0	59

Figure 3.6 Record Definition File.

The Key Definition File (KDF), is also key-sequenced and contains one record for each primary key and each alternate key declared for each record in the schema. The identification of the KDF record comes from the object number that has been assigned by the OBL. The RDF links to the KDF on RECORD-NUMBER, one to many and KDF links to the OBL by DEF-NUMBER/DEF ELEMENT one to one. This layout is shown in figure 3.7

RECORD NUMBER	KEY NUMBER	DEF NUMBER	DEF ELEMENT	KEYTAG VALUE	OFF-SET	SIZE
19	0	19	1			3
19	1	19	2			13
19	2	19	3			2
19	3	19	4			14
19	4	19	5			3
19	5	19	6			2
19	6	19	7			4

Figure 3.7 Key Definition File.

With the dictionary files described and examples provided, it is important to realize that these files can be

linked through primary and alternate keys. This capability provides the DBA with a complete picture of the data elements within the supply system. Figure 3.8 shows one set of possible linkages between the dictionary files. By no means does this diagram show all possible links between the files.

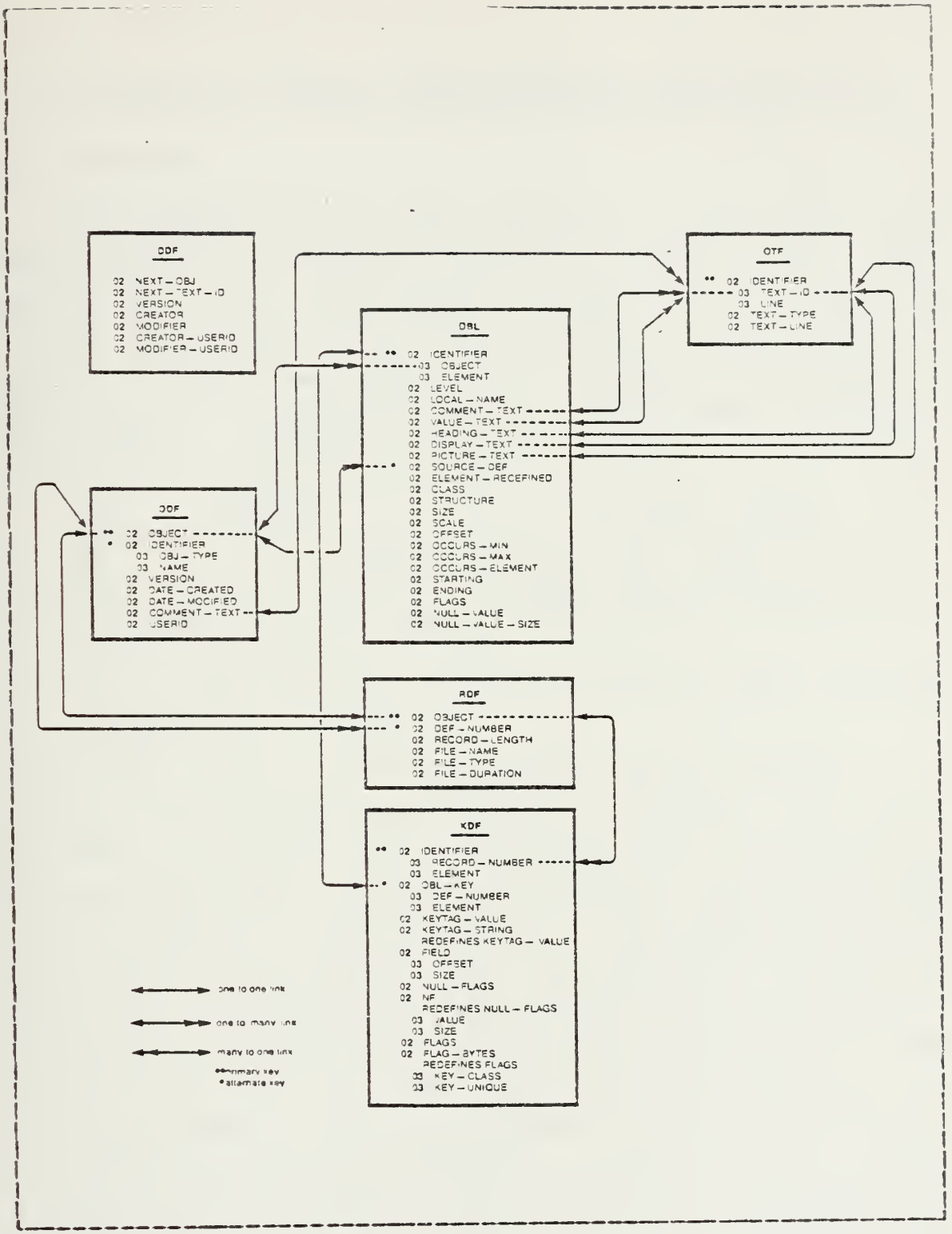


Figure 3.8 Dictionary Structure.

IV. DESIGN OF A SPLICE DIRECTORY BASED UPON TANDEM DBMS

A. DIRECTORY

The software package that formulates the data dictionary does a good job of describing each data element (i.e., tells what it is) but it does little to tell a user the location of each data element (i.e., where it is) within the SPLICE system. Without the capability of retrieving this directory information, the present layout of the data dictionary cannot satisfy the following queries:

- What locations in the network stock item 5905-00-255-3699, resistor, fixed, composition?
- Who is the inventory manager who holds technical responsibility and what is the latest status for a particular electronic tube?
- Who is the manufacturer of an item and where is this manufacturer located?

Even though all of these questions can be answered in a matter of time by any of the UADPS-SPs manually, they cannot be answered on-line by use of the proposed SPLICE system, the ENCOMPASS DBMS, or the data dictionary. The development of a "customized" data directory, to be defined using the DDL offered by the TANDEM DBMS, is suggested. Basically this would be the same as defining a database through the capabilities of the DDL. Once this directory has been created and is in use, the data dictionary itself can be queried by ENFORM to obtain information about the directory's data structure and location. Creation of a simple directory is the foundation of the concept. As technology improves and the directory becomes a more useful software tool, its capabilities no doubt will be expanded.

The directory which is to be defined is one which contains the necessary information that will answer the top ten or fifteen questions at a local stock point. These questions are most likely to be about line items (repairables or consumables), their location within the system, quantity on hand, manufacturer, and who has control over the items within the Navy Supply System. To answer these questions the directory must store information about the basic aspects of line items within the system. Line items will range from repairables for ships, airplanes, and machinery to consumables such as screws, bolts, and clothing. Therefore it is assumed that this directory will contain information about inventories and locations.

The TANDEM DBMS uses a relational approach in defining a database and that fits our needs well. The relational approach to data is based on the realization that files that obey certain constraints may be considered as mathematical relations, and hence that elementary theory about mathematical relations may be brought to bear on various practical problems of dealing with data in such files [Ref. 15].

B. DATA ELEMENTS FOR THE DIRECTORY

The files within the directory must store information on several aspects of the line items: stock item identification, location of manufacturer, and stock point location. The information describing each of these areas of the supply system is shown in table IX, table X, and table XI.

C. FILE STRUCTURES

As mentioned earlier, all the files in the directory are defined as relational files that are uniquely defined by the value stored in its primary key field. With a primary key value assigned, ENSCRIBE, the database manager of TANDEM,

TABLE IX

Stock Item Identification

• NATIONAL STOCK NUMBER	* ACCOUNT/COGNIZANCE
• MANUFACTURER CODE NUMBER	* NOUN NAME
• PART NUMBER	* UNIT OF ISSUE
• SPECIAL MATERIAL ID CODE	* SITE NAME
• LOCATIONS	* ITEM MANAGER
• TECHNICAL RESPONSIBILITY	* PRICE

TABLE X

Manufacturer File

• MANUFACTURER CODE NUMBER	* ORDER NUMBER
• MANUFACTURER NAME	* ORDER DATE
• LOCATIONS	* EST DELIVERY DATE
• PARTNAME	* PART NUMBER
• SALES PERSON	* PHONE NUMBER

can randomly position itself anywhere within these files for a read, write, or update operation. Another step in defining each file is to evaluate what data elements other than the primary key will frequently be used as access paths into the files. If these data elements are assigned as alternative key fields, then ENSCRIBE can selectively read

TABLE XI

Stock Point Location File

• NATIONAL STOCK NUMBER	* SITE NAME
• QUANTITY ON HAND	* UNIT OF ISSUE
• LOCATION	* PRICE
• REORDER POINT	* BACKORDER
• OUTSTANDING REQUISITIONS	* SAFETY LEVEL
• SUPPLY CONDITION CODE	* REPAIR STATUS
• PURPOSE CODES	* MODE OF SHIPMENT
• HOLD CODE	* PERSONNEL DIRECTORY

records from the files on the basis of either full or partial alternative key values. These keys are not required to be unique [Ref. 13].

1. Stock Item Identification

The Stock Item Identification information is broken into three files: line items, stock location, and material control. The data elements within these files describe what line items are stocked in the Navy Supply System, where these items are stocked, and who has been assigned the overall responsibility of each individual line item. Figure 4.1 shows the information in each file, primary and alternative keys and the linkage between the files.

2. Manufacturer File

The manufacturer file uses three files identified as: manufacturer, order, and order description. These files hold information pertaining to the manufacturer

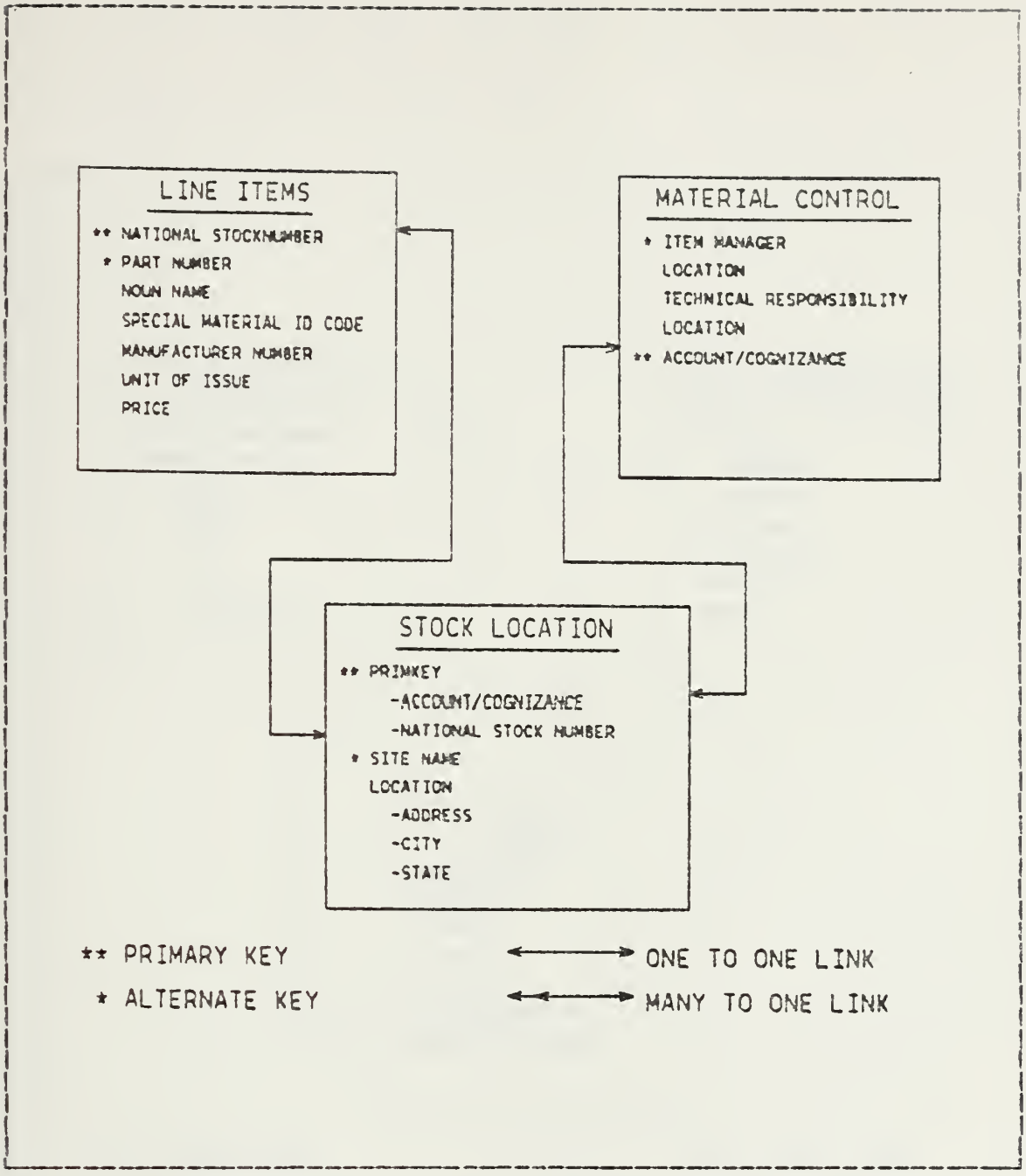


Figure 4.1 Stock Item Identification.

identification and location, orders outstanding with that manufacturer, and when these orders are to be delivered. They will also show what part these orders are for, how much

they cost, and with whom this order was placed. Figure 4.2 represents this structure with the keys identified plus the relationships between these files.

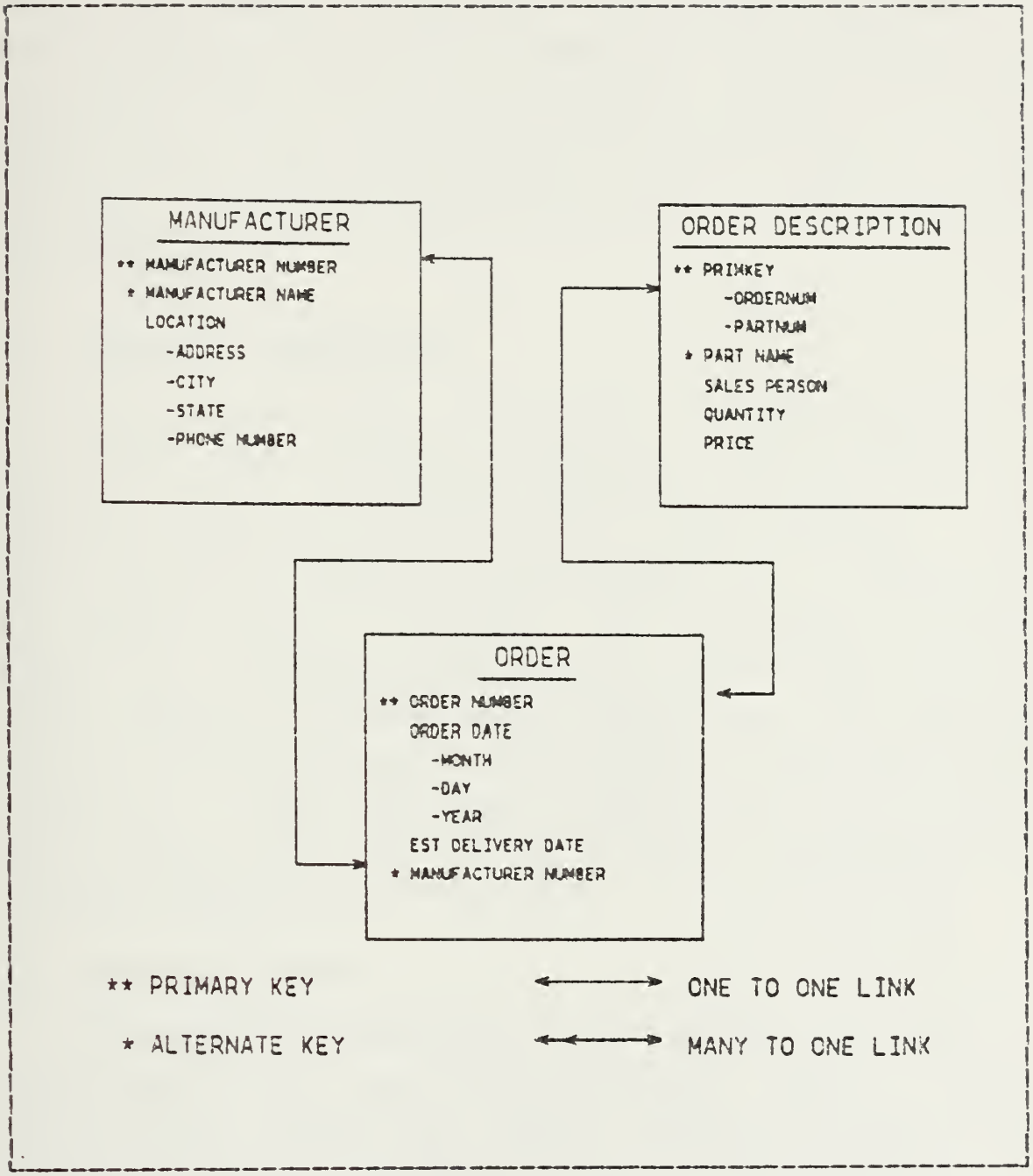


Figure 4.2 Manufacturer File.

3. Stock Point Location File

The information needed in this file is divided into three files that are labelled inventory detail, site directory, and status. These files would give the requestor detailed information on each stock item that is carried by this particular activity, a directory of the personnel within the activity stock point, and the status of any documents of the requestor that might have been passed to this activity. Figure 4.3 shows these files and how they relate to one another.

D. DIRECTORY DISTRIBUTION

The question of which distribution configuration to use in locating this directory within the SPLICE project can best be handled by deciding the characteristics of the data/information to be controlled by the directory. The data within the defined SPLICE directory can be classified as either static or dynamic when considering updates [Ref. 16]. The static classification does not mean the data cannot change but changes occur infrequently and therefore this type of data presents few problems in controlling data integrity. On the other hand, dynamic data, that is data with a low to high update rate, presents a challenge when trying to maintain data integrity.

E. STATIC OR DYNAMIC

The SPLICE directory that has been defined consists of a combination of data that has static/dynamic updates. Within this area of concern are several classes of data. The classes are divided according to the complexity of the

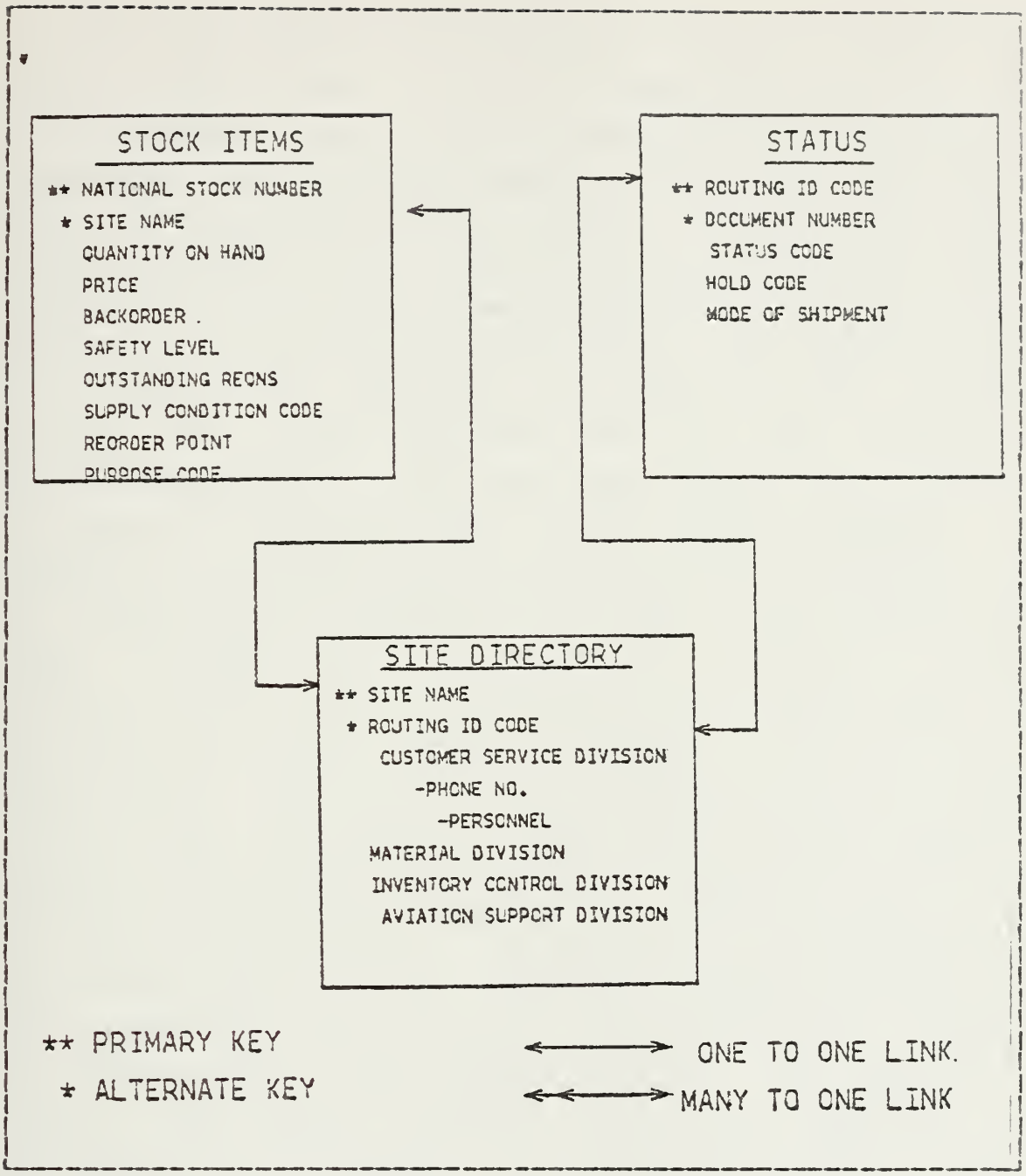


Figure 4.3 Stock Point Location File.

updates. These classes of updates range from CLASS 0 (i.e., data which does not change) tc CLASS 4 (i.e., data for which an update may trigger an action in a different machine) [Ref. 16].

The data that would be located in the stock item identification files meets the definition of static updates and contains CLASS 1 data. Updates to the data in this portion of the directory would be made by additions, deletions, or replacement and would most likely be made via batch mode. Simple updates of data such as these, if performed twice, cause very little harm. The data in the manufacturer files would be grouped in the dynamic category with updates being fairly low and containing CLASS 2 data. This class of data is a little more sensitive to updates and any updates applied twice could cause the loss of data integrity. Update transactions in this area would involve cancellations of orders or possible changes in the quantity ordered. If repeated, this would mean the loss of data integrity. If the update is only delayed but not lost, there would be no harm. A simple control technique of assigning a serial number to each update transaction would ensure that no transaction is lost or double-processed. The stock point location data is also dynamic in nature but with a high update rate. The data is categorized as CLASS 3 meaning that the data is very sensitive to time-critical updates. Updates in this area, if reapplied at a different time, may not have the same effect. For example, if the quantity-on-hand field shows 100 EA in stock for a particular screw but the recent issue of 50 EA is not reflected before a request for 75 EA comes in from another activity, the requestor is going to stop looking since he thinks his demand can be satisfied by this activity. In actuality he needs to find an activity that has 25 EA more. A control technique that can be applied is the use of timestamps to ensure that transactions are not processed in an invalid sequence. Serial numbers may be needed as well as timestamps to prevent loss or double processing of transactions on recovery [Ref. 17].

The concept of system architecture and the network database management system software deciding the distribution strategies for dictionaries also holds true for the distribution of directories. The four types of distribution strategies which where discussed in Chapter III, along with their advantages and disadvantages, similarly apply to the directory portion of the DDS. Of the four strategies mentioned, it is suggested that the hybrid solution be given the most attention though its inherent complexity is acknowledged.

F. HYBRID SOLUTION

A hybrid solution is one where multiple copies of the subsets of the directory are replicated within the network and each node may have a partitioned fraction of the directory [Ref. 16].

With this description in mind it is suggested that the stock identification files and the manufactureer files be replicated at the inventory control points located at ASO PHILADEPHIA and SPCC MECHANISBURG plus NSC NORFOLK, NSC OAKLAND, and NSD SUBIC BAY. The stock point location files would then be be partitioned at each UADPS-SP activity that held line items for issue. The justification behind this type of distribution falls back to the characteristics of the updates toward each file, and the differences in data at each site. Since the first two files will experience static or low volume updates and therefore is less likely to have high volume of communication, the problems concerning reliability, response time for both retrievals and updates, and data integrity are not as hard to control or manage. In the case of the stock point location files there is very high updating and the make-up of line items carrried at each site is very diversified. Since the data that pertains to each

dispersed activity is centralized at that location the problems of data integrity, communication cost, and synchronization cost should be lowered.

This solution takes advantage of natural patterns of usage and transaction processing and is structured around those patterns. For example, most of the customers at a local site will normally operate through their local supply department to meet their needs. These transactions require only simple transaction processing. This makes it feasible to keep the stock point location files at each local site. It is also true that sometimes the customer cannot be satisfied by the local site and therefore has the need to go off station which will require a complex transaction to be performed. When this happens the requestor is looking for a central site that can give directions to a site that can handle the requirement. The directions that are being sought will be handled by the replicated directory at whichever of the five sites is closest.

This information is obtained by a user through the use of a "LINK" command statement that is offered by ENFORM, the query specification language. This statement offers a convenient means of connecting associated data independent of its location and without altering physical files. Using a key field, LINK identifies the relationship between two or more file descriptions from the directory. Once this process has taken place the relationships are combined so that they appear to the user as a single logical file [Ref. 12]. The three groups of files that have been previously defined are shown together in figure 4.4. This figure shows some of the possible links between the various distributed files.

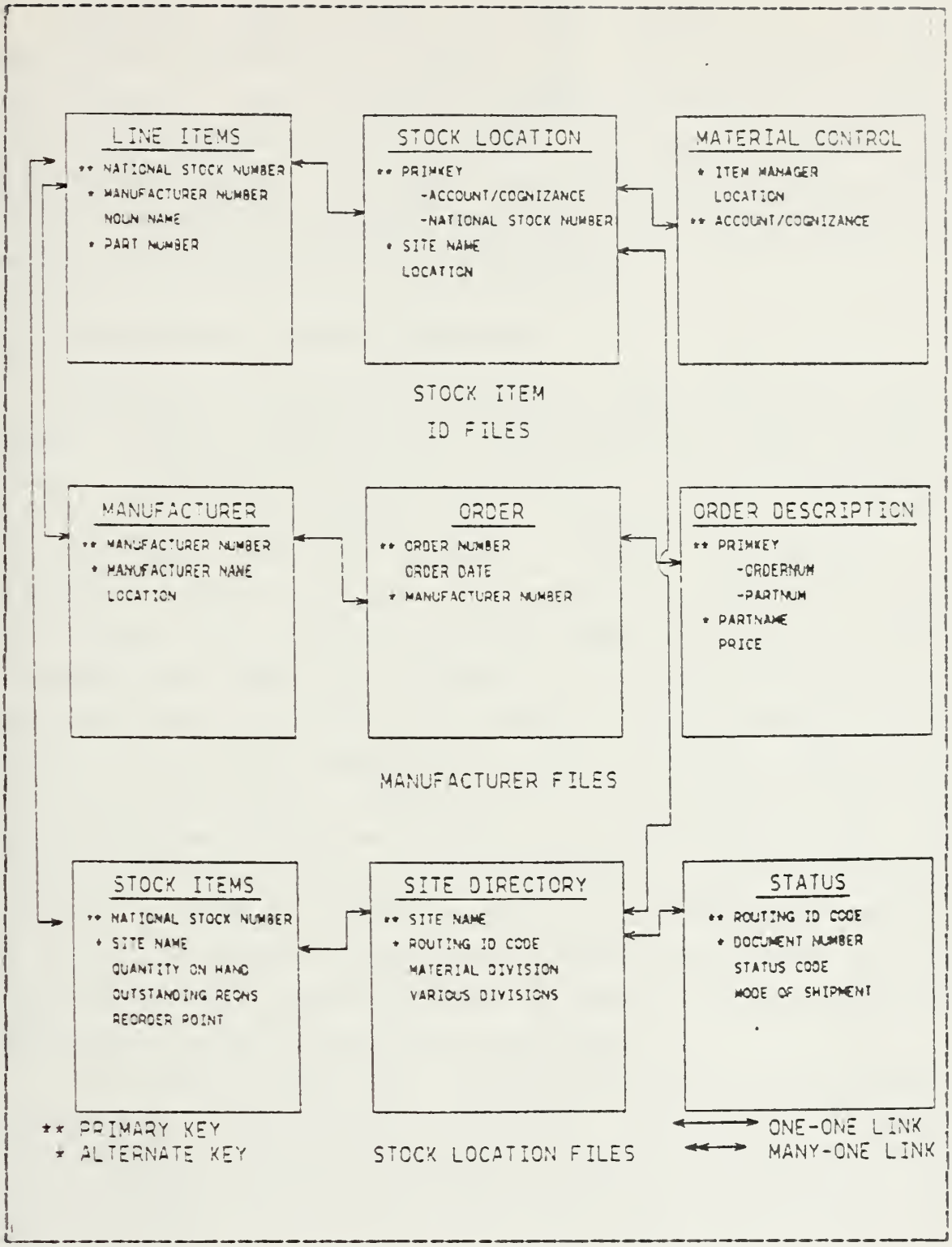


Figure 4.4 SPLICE Directory.

The Navy Supply System is a good example of a complex distributed information system. To solve a problem within this system, the patterns of usage and methods of transaction processing must be followed and used to answer the questions of where, how much, and what data to put at each node. The hybrid solution does this, thus facilitating the design and implementation of a directory.

G. BENEFITS OF A SPLICE DIRECTORY

The SPLICE directory as defined above would put thousands of useful records at a wide variety of users' fingertips. Through this on line directory, a user could access the location of available stock, manufacturer's action on orders, and even the status of their requisitions at different sites. Valuable time would no longer be wasted looking up and writing down line items carried by a site, manufacturer's name and address, or stock site personnel contacts every time they were needed. The SPLICE directory would provide this information to the requestor. The directory records would be available through any of the sixty-two proposed sites, would be current, and updatable. Once the directory information has been retrieved, a user can revise, create, or delete a report/record of the selected information that is needed by using the features of ENFORM. At the same time, the directory would be secure by only allowing alterations made by authorized personnel.

It is easy to see that the directory could provide many benefits to a wide variety of users. The immediate result of providing a directory would be that the directory would take the place of numerous manual searches, saving time and money.

V. CONCLUSIONS

A. IDENTIFICATION OF NEED

Considering the diversity of the application programs that will be tied to UADPS-SP under the SPLICE project, there is a critical need for a tool to manage the preponderance of system resources. A DDS will offer the capabilities needed to control and manage the data resources in the intricate and highly complex distributed SPLICE network system. Relative to this, a key aspect of a DDS is that it provides resource independence.

B. BENEFITS

There are many benefits that a SPLICE DDS can provide. The DDS serves DBAs, system analysts, software designers and programmers by requiring definitions of system data elements, files, programs and reports. Conversion to uniform resource description standards allows systems within the SPLICE network to interface easier and more freely. By establishing standards of data definitions and descriptions for applications programs throughout the SPLICE system, the DDS serves as the focal point for future analysis and design. Using a DDS, data can be managed to the best advantage for the existing information system and can be effectively redeployed to meet changing information requirements. The DDS can be used to generate test data and check results before allowing changed programs to replace the production versions.

C. SYNOPSIS

This thesis reviewed the current status of the SPLICE DDS and proposed a TANDEM DDS for SPLICE. Basic concepts and considerations in evaluating a DDS were discussed and these fundamental principles were applied to the TANDEM Data Dictionary and a preliminary Directory design based upon the TANDEM DBMS. A brief comparison was made of the TANDEM Data Dictionary and other commercial products.

The TANDEM Data Dictionary is essentially a data element dictionary. It is a DBMS-application approach to a DDS. The product does not have directory capabilities at this time. The dictionary is an active/dependent system which would require that SPLICE system components depend upon the dictionary for their data. Its usefulness, in its present form, would primarily be as a tool for manipulating information about data elements in a database. The TANDEM dictionary operates effectively as a conversion facility for three different programming languages as well as for the DDL compiler. By placing the data definitions and data structures in a single language, the DBA gains control over the database design and implementation. Considering the volume of data in SPLICE and its complexity, this would be an advantageous feature.

D. FINAL COMMENT

A DDS is a powerful information management tool with potential, positive payoff for the SPLICE project. It is important to note that the DDS can only improve system productivity and accuracy if its design potential is explcited. As a minimum, it must indicate what and where data is stored, when and how updates are handled and which applications programs are accessed. A DDS implemented to perform these functions clearly provides a great deal of automated system control.

Naturally there are some drawbacks in employing a DDS. A DDS implies system overhead and maintenance as well as an initial burden on database management personnel. However, proper planning and early organization-wide commitment to managing data as a resource will create a climate in which an active DDS can facilitate orderly data processing while avoiding chaotic data management. A DDS can be a valuable tool in the management of distributed data for the SPLICE project.

LIST OF REFERENCES

1. Fleet Material Support Office, Department of the Navy Document No. F94L0-001-9260-FD-SU01, Stock Point Logistics Integrated Communications Environment (SPLICE), Functional Description, 1 May 1980.

2. U.S. Navy Fleet Material Support Office, Environment Division: Code 9441, Stock Point Logistics Integrated Communications Environment (SPLICE), Software Design, 19 March 1979.

3. Naval Supply Systems Command, Stock Point Logistics Integrated Communications Environment (SPLICE), Hardware Configuration Management Plan (HCMP), 1 November 1980.

4. Naval Postgraduate School Report NPS-54-82-003, Functional Design of a Local Area Network for the Stock Point Logistics Integrated Communications Environment, by N.F. Schneidewind, December 1982

5. National Bureau of Standards Special Publication 500-3, Technical Profile of Seven Data Element Dictionary/Directory Systems, February 1977

6. Allen, F.W., Loomis, M.E.S., and Mannino, M.V., "The Integrated Dictionary/Directory System," Computing Surveys, June 1982.

7. Naval Postgraduate School Report NPS54-83-015, A Distributed Operating System Design and Dictionary/Directory for the Stock Point Logistics Integrated Communications Environment, by N.F. Schneidewind and D.R. Dolk, November 1983.

8. Data Dictionary Systems Working Party (1977), "The British Computer Society Data Dictionary Systems Report," SIGMOD Record, vol. 9, no. 4, December 1977

9. Teichroew, D. and Hershey, E., "PSL/PSA: A Computer-aided Technique for Structured Documentation and Analysis of Information Processing Systems," IEEE Transactions on Software Engineering, Vol SE3-No.1, January 1977, 41-48.

10. Fleet Material Support Office, Department of the Navy, Document No. F94L0-9260-001-SS-SU01, Stock Point Logistics Integrated Communications Environment (SPLICE), System Specification, 2 February 1981.

11. TANDEM Computers Incorporated, Enscribe (TM) Programming Manual, April 1981

12. TANDEM Computers Incorporated, Enform (TM) Users Guide, October 1982

13. TANDEM Computers Incorporated, Data Definition Language (DDL) Programming Manual, December 1981

14. Lefkovits, H.C., Sibley, E.H. and Lefkovits, S.L., Information Resource/Data Dictionary Systems, QED Information Sciences, Inc., 1983

15. Krcenke, David, Database Processing, Science Research Associates, Inc., 1977

16. Martin, James, Design and Strategy for Distributed Data Processing, Prentice-Hall, Inc., 1981

17. Pooch, U.W., Greene, W.H. and Moss, G.G., Telecommunications and Networking, Little, Brown & Company (Canada) Limited, 1983

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22134	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Naval Postgraduate School Computer Technology Curriculum Office Code 37 Monterey, California 93943	1
4.	Assistant Professor Dan R. Dolk, Code 54Dk Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943	1
5.	Professor Norman F. Schneidewind, Code 54Ss Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943	3
6.	Commander Tom Pigoski, Code G-30D Naval Security Group 3801 Nebraska Ave. NW Washington, D.C. 20390	1
7.	Lieutenant Colonel Joseph P. Mullane, USMC Code 0309 Marine Corps Representative Naval Postgraduate School Monterey, California 93943	1
8.	Commandant of the Marine Corps (Code CC) Headquarters Marine Corps Washington, D.C. 20380	2
9.	Lieutenant David C. Ruff, SC, USN Route 6 Tupelo, Mississippi 38801	1
10.	Captain James R. Johnson, 534-60-9291/3002 USMC Headquarters Marine Corps (MCC 009) Washington, D.C. 20380	1

297541

Thesis
R8442 Ruff
c.1 A preliminary DDS
design for SPLICE based
upon the TANDEM DBMS.